数据结构实验报告四

实验名称：查找和排序算法的实现

### 实验目的及实验要求

1. 学生在实验中体会各种查找和内部排序算法的基本思想、适用场合，理解开发高效算法的可能性和寻找、构造高效算法的方法。
2. 掌握运用查找和排序解决一些实际应用问题。

### 2实验内容及实验步骤（附运行结果截屏）

1. 编程实现一种查找算法（如折半查找、二叉排序树的查找、哈希查找等），并计算相应的ASL。
2. 编程实现一种内部排序算法（如插入排序、快速排序等）。

实验步骤：

本次实验不涉及自己完成某种数据结构的编码问题，因此在掌握了相应的算法后可以很快的编写代码完成既定目标。搜索选材为《I have a dream》，使用了哈希查找的方式，时间复杂度约为。

排序放大选择了快速排序并进行了编码。

核心代码：

//HashSearch.h 哈希查找

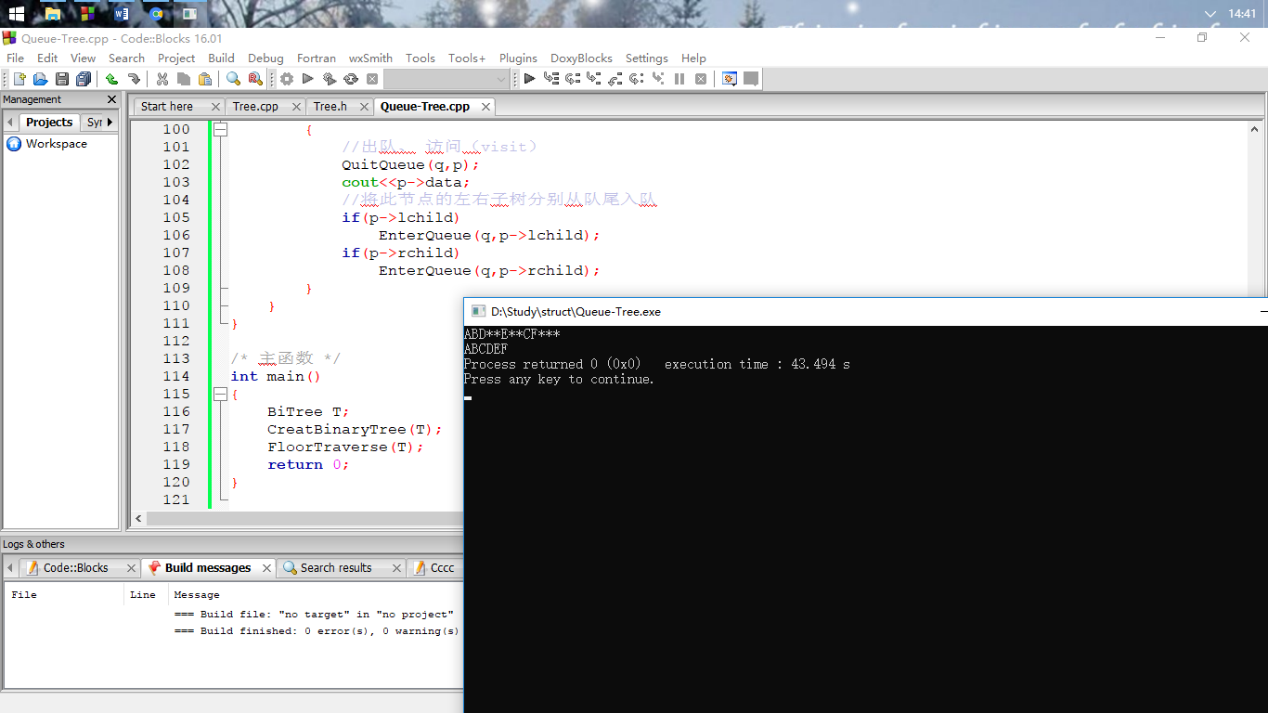
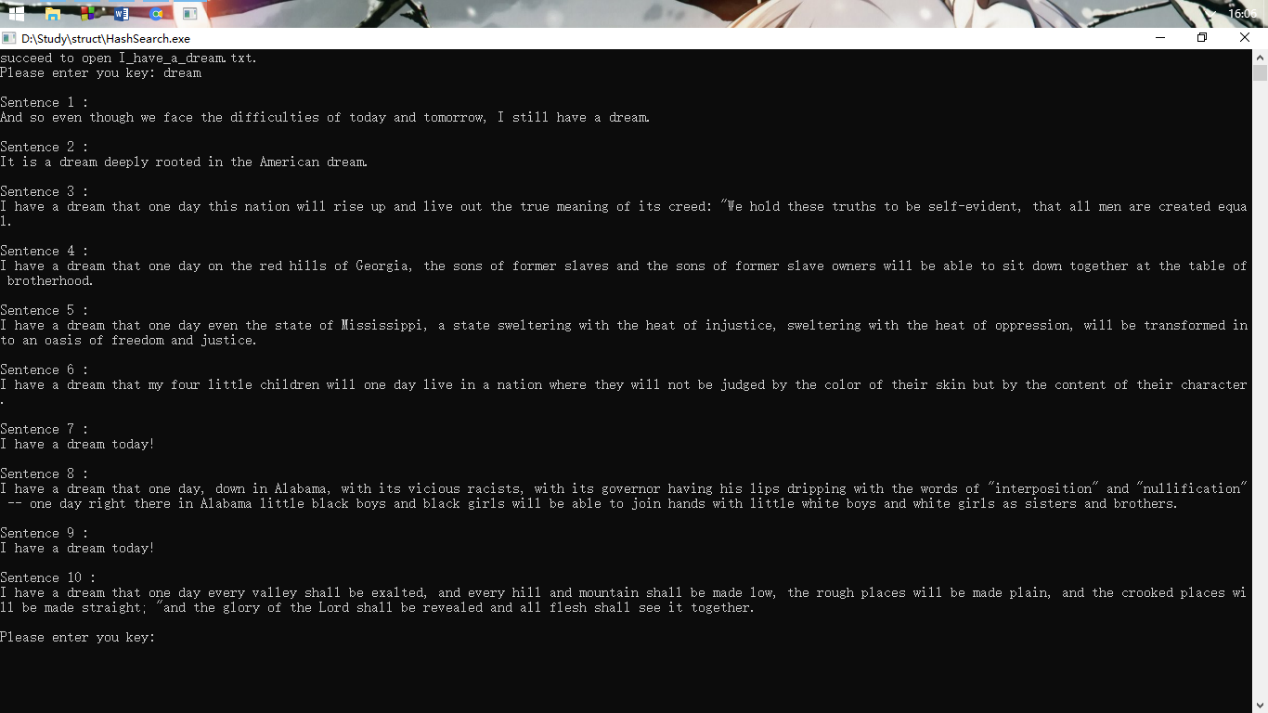
//生成哈希表和搜索

1. void CreatHashTable() //生成哈希表
2. {
3. hash\_chain=new hash\_node [HASH\_SIZE];
4. for(int i=0; i<HASH\_SIZE; i++) //初始化
5. {
6. hash\_chain[i].key="";
7. hash\_chain[i].number=0;
8. hash\_chain[i].next=NULL;
9. }
10. int pos;
11. string temp;
12. for(int i=0; i<container.size(); i++)
13. {
14. temp=container[i];//去除结尾符号
15. transform(temp.begin(),temp.end(),temp.begin(),Cut);
16. //cout<<container[i]<<endl;
17. istringstream iss(temp);
18. while(iss>>temp) //读取单词
19. {
20. //cout<<temp<<endl;
21. pos=Hash(temp);
22. hash\_node\* p1=&hash\_chain[pos];
23. if(p1->key=="")
24. {
25. p1->key=temp;
26. p1->number=i;
27. }
28. else
29. {
30. bool repeat=false;
31. while(p1->next!=NULL)
32. {
33. if(temp==p1->key&&i==p1->number)
34. {
35. repeat=true;
36. break;
37. }
38. p1=p1->next;
39. }
40. if(temp==p1->key&&i==p1->number)
41. {
42. repeat=true;
43. }
44. if(repeat)
45. continue;
46. hash\_node\* pnew=new hash\_node(temp,i,NULL);
47. pnew->next=p1->next;
48. p1->next=pnew;
49. }
50. }
51. }
52. }
53. hash\_node::hash\_node(string a,int b,hash\_node\* c) //结构体构造函数
54. {
55. key=a;
56. number=b;
57. next=c;
58. }
59. void SearchFor(string word) //搜索函数
60. {
61. int pos=Hash(word),k=1;
62. hash\_node\* p1=&hash\_chain[pos];
63. while(p1!=NULL) //遍历链表
64. {
65. if(p1->key==word)
66. {
67. cout<<endl;
68. cout<<"Sentence "<<k++<<" :"<<endl;
69. cout<<container[p1->number]<<endl;
70. }
71. p1=p1->next;
72. }
73. cout<<endl;
74. }

//QuitSort.h 快速排序

1. int getPartion(int \*array, int low, int high)
2. {
3. int key = array[low];
4. while (low < high)
5. {
6. while (low < high && key <= array[high])
7. high--;
8. swap(array[low], array[high]);
9. while (low < high && key >= array[low])
10. low++;
11. swap(array[low], array[high]);
12. }
13. return low;
14. }
15. void QuitSort(int \*buf,int low,int high)
16. {
17. if (low < high)
18. {
19. int key = getPartion(buf, low, high);
20. QuitSort(buf, low, key - 1);
21. QuitSort(buf, key + 1, high);
22. }
23. }

截图：



3 实验体会（实验遇到的问题及解决方法）

本次实验练习了文件操作，这一块儿有所遗忘但看一遍书后迅速再次掌握并顺利地完成了实验。由于算法不是太复杂而且之前在课上老师已经介绍地很完备课下也及时跟进复习了所以并没有在算法上遇到太多问题，整个过程很顺利。